DS Info - Corrigé :

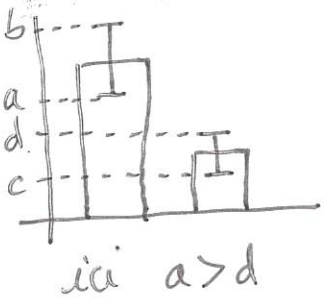
Exercice 1

1)
```
def moy(L):
    return sum(L)/len(L)

def ecart(L):
    m = moy(L)
    S = 0
    for x in L:
        S = S + (x-m)**2
    v = S/len(L)
    return v**(1/2)
```

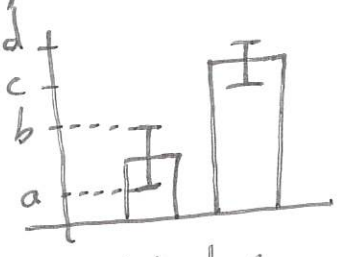
2)
```
def barres(T, C):
    mT = moy(T)
    mC = moy(C)
    sT = ecart(T)
    sC = ecart(C)
    return (mT-sT, mT+sT, mC-sC, mC+sC)
```

3) Pour que les barres d'erreur $[a,b]$ et $[c,d]$ ne se recoupent pas, il y a 2 situations possibles :



ici $a > d$

ou



ici $b < c$

Ainsi les barres d'erreurs ne se recoupent pas si et seulement si

$a > d$ ou $b < c$

d'où le code :

```
def impact_engrais(T, C):
    (a,b,c,d) = barres(T, C)
    return (a>d) or (b<c)
```

Exercice 2

1) `L_t = [i*h for i in range(Nb_iter + 1)]`

2) ligne 14 :

`h = T_max / Nb_iter`

lignes 22 et 23 :

`N_next = N + h*(r*N - C*N*P)`
`P_next = P + h*(-m*P + g*C*N*P)`

3)
```
import matplotlib.pyplot as plt
plt.plot(L_t, L_N)
plt.plot(L_t, L_P)
plt.show()
```
} pour la figure 1

```
plt.plot(L_N, L_P)
plt.show()
```
} pour la figure 2

4) Pour que l'approximation soit meilleure, il faut que le pas h soit plus petit, ou encore que le nombre d'itérations Nb_iter soit plus grand. On peut donc modifier la ligne 13 et choisir par exemple `Nb_iter = 10000`.

Exercice 3

1)
```
import random as rd
def bernou(p):
    t = rd.random()
    if t <= p:
        return 1
    else:
        return 0
```

```
def binom(n,p):
    S = 0
    for k in range(n):
        S = S + bernou(p)
    return S
```

2)
```
def maximum(L):
    maxi = L[0]
    for x in L:
        if x > maxi:
            maxi = x
    return maxi
```

3)
```
def maxi_alea(N,n,p):
    L = [binom(n,p) for k in range(N)]   # L contient [X_1, X_2, ..., X_N]
    return maximum(L)
```

4) Commençons par calculer la loi de Y c'est-à-dire par construire la liste L des $\mathbb{P}(Y=k)$ pour $k \in Y(\Omega) = [\![0,n]\!]$:

```
def loi(N,n,p):
    L = []
    for k in range(n+1):
        compt = 0
        for i in range(10000):
            if maxi_alea(N,n,p) == k:
                compt = compt + 1
        pk = compt/10000
        L.append(pk)
    return L
```

on estime ici la valeur de $p_k = \mathbb{P}(Y=k)$ en opérant 10000 fois l'expérience

On cherche ensuite la valeur de k telle que $L[k]$ soit maximal. Il s'agit d'utiliser une variante de la fonction maximum renvoyant l'indice du maximum d'une liste :

```
def indmax(L):
    ind = 0
    maxi = L[0]
    n = len(L)
    for k in range(1, n):
        if L[k] > maxi:
            maxi = L[k]
            ind = k
    return ind
```

Enfin on utilise la fonction indmax sur la liste donnant la loi de Y :

```
def maxi_proba_k(N, n, p):
    L = loi(N, n, p)
    return indmax(L)
```